Copyright

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.
About This Document

Scope
The scope of this document includes all the information required to install, configure and administer the SLEE application.

Audience
This guide was written primarily for system administrators and persons installing, configuring and administering the SLEE application. The documentation assumes that the person using this guide has a good technical knowledge of the system.

Prerequisites
Although there are no prerequisites for using this guide, familiarity with the target platform would be an advantage.

A solid understanding of Unix and a familiarity with IN concepts are an essential prerequisite for safely using the information contained in this technical guide. Attempting to install, remove, configure or otherwise alter the described system without the appropriate background skills, could cause damage to the system; including temporary or permanent incorrect operation, loss of service, and may render your system beyond recovery.

This manual describes system tasks that should only be carried out by suitably trained operators.

Related documents
There are no documents related to this document.
Document Conventions

Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Network Charging and Control (NCC) documentation.

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Special Bold</strong></td>
<td>Items you must select, such as names of tabs. Names of database tables and fields.</td>
</tr>
<tr>
<td><em>Italics</em></td>
<td>Name of a document, chapter, topic or other publication. Emphasis within text.</td>
</tr>
<tr>
<td><strong>Button</strong></td>
<td>The name of a button to click or a key to press.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>To close the window, either click <strong>Close</strong>, or press <strong>Esc</strong>.</td>
</tr>
<tr>
<td><strong>Key+Key</strong></td>
<td>Key combinations for which the user must press and hold down one key and then press another.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Ctrl+P</strong>, or <strong>Alt+F4</strong>.</td>
</tr>
<tr>
<td><strong>Monospace</strong></td>
<td>Examples of code or standard output.</td>
</tr>
<tr>
<td><strong>Monospace Bold</strong></td>
<td>Text that you must enter.</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td>Used to indicate variables or text that should be replaced.</td>
</tr>
<tr>
<td><strong>menu option &gt; menu option &gt;</strong></td>
<td>Used to indicate the cascading menu option to be selected, or the location path of a file.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>Operator Functions &gt; Report Functions</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><em>IN/html/SMS/HelpText</em></td>
</tr>
<tr>
<td><strong>hypertext link</strong></td>
<td>Used to indicate a hypertext link on an HTML page.</td>
</tr>
</tbody>
</table>

Specialized terms and acronyms are defined in the *Glossary* at the end of this guide.

Terminology

This topic explains any terminology specific to this manual.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS Customer</td>
<td>ACS customers are set up in the ACS Customer screen. They configure systems to provide services to subscribers.</td>
</tr>
<tr>
<td>Service Provider</td>
<td>CCS service providers are the same as ACS customers and are set up in the ACS Customer screen. There is additional service provider configuration provided in CCS.</td>
</tr>
<tr>
<td>Customer</td>
<td>Customers in CCS refer to the customers configured on the Subscriber Management screen.</td>
</tr>
<tr>
<td>Subscriber</td>
<td>A subscriber account is set up for each MSISDN which uses services provided by the service provider.</td>
</tr>
</tbody>
</table>
Chapter 1
System Overview

Overview

Introduction

This chapter provides a high-level overview of the application. It explains the basic functionality of the system and lists the main components.

It is not intended to advise on any specific Oracle Communications Network Charging and Control (NCC) network or service implications of the product.

In this chapter

This chapter contains the following topics.

Introduction to the Service Logic Execution Environment 1
Main Components of the SLEE 2
SLEE Interfaces 5
Application Programming Interface 6

Introduction to the Service Logic Execution Environment

Introduction

The Service Logic Execution Environment (SLEE) provides a common execution environment for existing Oracle NCC products, including:

- Advanced Control Services (ACS)
- Charging Control Services (CCS)
- Voucher and Wallet Server (VWS)
- Messaging Manager (MM)

It provides mechanisms for multiple interfaces to communicate events with the call, therefore simplifying the service logic interfaces.

Functionality overview

The SLEE provides the following functionality:

- SLEE process monitoring/restart
- Simultaneous management of multiple different service logic applications.

Functionality for hosted services

The main functions of the SLEE are to provide hosted services with the following functionality:

- Event handling/call matching
- Event scheduling
- Call thread and context data control
Service logic support

The SLEE implements the common components of service logic within a single environment. It provides the following to service logic interfaces:

- A well-defined, open interface for the handling of call control threads, call context data and application management
- Efficient flexible mechanisms for multiple interfaces to communicate events with the call.

The SLEE maintains integrity and ensures high performance when managing multiple messages from multiple underlying networks to multiple applications.

Example setup

This figure shows a typical SLEE setup.

Main Components of the SLEE

Introduction

The NCC SLEE uses the following components.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLEE.cfg</td>
<td>This file holds the main configuration for the SLEE and startup.</td>
<td>Configuring the SLEE (on page 10)</td>
</tr>
<tr>
<td>SLEE shared memory</td>
<td>Shared memory used by SLEE applications and interfaces.</td>
<td></td>
</tr>
<tr>
<td>SLEE API</td>
<td>Application Programming Interface used by SLEE applications and interfaces.</td>
<td></td>
</tr>
<tr>
<td>watchdog</td>
<td>The SLEE watchdog monitors SLEE applications and interfaces, restarting them</td>
<td>watchdog (on page 16)</td>
</tr>
</tbody>
</table>
For a full description of each component, refer to the topics below.

**SLEE shared memory**

The `/tmp/slee` is the default file used by all SLEE processes at start up to get the shared memory key of the SLEE shared memory. This file must exist and must be the same for all processes wishing to access the same shared memory.

If this file is removed, a SLEE process which is starting up will fail to find shared memory and will exit. If this file is removed, you must restart the SLEE.

For information about overriding the location and filename by setting the environment variable `SLEE_FILE` for all SLEE processes, see *Optional environmental variables* (on page 10).

**Entity relationship**

This diagram shows how entries in the `SLEE.cfg` file are related.
Applications

An example of an application is a piece of service logic that provides local number portability processing. An application may support multiple services (for example, it may provide voice VPN in addition to number portability). The SLEE can support multiple different applications simultaneously.

An application is a program that provides a specific set of services to the interfaces. The SLEE allows multiple copies of an application to be started, to enable performance advantages in SMS environments and in cases where an application can block (very briefly) during execution of service logic. This may occur, for example, during an Oracle database read.

Note: Care must be taken to avoid applications from blocking.

Under ideal conditions, an application would never communicate directly with external entities. However, in some cases, this cannot be avoided.

Applications are provided with a set of classes and objects, which provide an interface to the SLEE. All the API functionality for an application is based around call instances. A call instance must be created by an interface (or application acting like an interface). The application’s call context memory may also be allocated via the API that is associated with the call instance.

Application instance

Each executing copy of the application is known as an application instance. The SLEE can support multiple running instances of each application program. This is useful in multi-processor environments, where it is possible to be processing events for more than one call in parallel.

Service

Each application may provide multiple services. In the above example, we have an application that provides service and local number portability (SNP & LNP). That application would therefore have two services defined - one for each SNP and LNP.

Service handles

The service handle is defined in the service entity. When a new call is presented to the application, the service handle indicates the particular service for which the call is intended.

Service keys

The service key is the SLEE’s mechanism for providing service discrimination. Each service key maps to a service or interface.

Note: A service key is a generic name for the identifiers for a service or interface. Service keys may be derived in some cases from the INAP InitialDP service key.

For more information about configuring service keys, see Service keys (see "SERVICEKEY" on page 18).

Interfaces

Each service may include an interface to a protocol, which is used to talk to an external entity (SSP, HLR, billing engine, alarm system).

An interface may also generate new calls for the services, applications and application instances.

Examples: Interfaces include:

- TCAP/SS7
- Billing Access
Interface handles are used by other elements within the SLEE (other interfaces or application instances) to identify the associated interface.

**SLEE tools**

The SLEE package also includes a set of tools for managing the SLEE. This table describes these tools.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>slee.sh</td>
<td>This script starts the SLEE.</td>
<td>Starting the SLEE.</td>
</tr>
<tr>
<td>stop.sh</td>
<td>This script shuts down the SLEE and clears the SLEE shared memory.</td>
<td>Stopping the SLEE.</td>
</tr>
<tr>
<td>clean</td>
<td>The clean utility is used to remove SLEE shared memory and semaphores after a unclean SLEE shutdown has occurred.</td>
<td>Removing Shared Memory.</td>
</tr>
<tr>
<td>check</td>
<td>This tool provides reports on SLEE attributes.</td>
<td>check (on page 25).</td>
</tr>
</tbody>
</table>

**SLEE Interfaces**

**Introduction**

An interface is a process that converts messages or events from outside the system from an external format into a common SLEE format. These may then be passed between elements of the SLEE. It also converts SLEE format messages to the external format required. The common format is an object representation of the event called a SLEE Event.

Interfaces are provided with a set of classes and objects, which provide an interface to the SLEE. Interfaces are the main sources of calls requiring processing. This is done by call instances and dialogs with those call instances. A dialog allows messages to be routed between call instances and interfaces. A dialog also allows identification of the call instance or dialog that a SLEE Event can support.

In addition to converting to and from SLEE events and external format events, the interfaces must also respond to SLEE management events. SLEE management events are a specific type of SLEE event.

**Types of interface**

There are different types of interface. Any interface can be one or more of these types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Can generate new call instances and dialogs due to external events.</td>
</tr>
<tr>
<td>Server</td>
<td>Can only respond to entities that create dialogs with it and make requests (that is, it can only respond to existing call instances).</td>
</tr>
<tr>
<td>Sink</td>
<td>A service that never sends a SLEE event. A special method of sending events to sinks is provided that does not use a dialog.</td>
</tr>
</tbody>
</table>
Interface communication through dialogs

While it is normal for interfaces to only communicate through dialogs belonging to call instances, it is also possible for an interface to talk to another interface, through a dialog. It is also possible for an application to talk to another application, through a dialog and call objects.

Example interfaces

This table describes some of the interfaces that may be supplied with the SLEE package.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>timerIF</td>
<td>The Timer interface interacts with the system’s real-time clock to provide time-out events to the service logic on demand.</td>
<td>Server</td>
</tr>
<tr>
<td>alarmIF</td>
<td>The Alarm interface interacts with the system error logging functionality to report alarms passed from the service logic application.</td>
<td>Sink</td>
</tr>
<tr>
<td>statsIF</td>
<td>The Statistics interface maintains and reports statistics to the management system. Statistics are effectively peg counts. statsIF requires a statistics.bin file to run. For more information about creating a statistics.bin file, see sfVerify (on page 27).</td>
<td>Sink</td>
</tr>
<tr>
<td>replicationIF</td>
<td>The Database Replication interface is an update requester. It processes updates to the system which are generated during call processing. For example, when a client uses call plan functionality to change their profile data. The replicationIF sends an update request to the smsMaster on an SMS. The SMF database is updated with the new information, and is then replicated to all other nodes. For more information about update requests and replication, see SMS Technical Guide.</td>
<td>Server/Sink</td>
</tr>
<tr>
<td>cdrIF</td>
<td>The CDR interface writes cdr files containing data received from SLEE CDR events.</td>
<td>Server</td>
</tr>
</tbody>
</table>

Note: Different installations use different SLEE interfaces. If the interface is not listed in the SLEE.cfg file, it is not being used by your installation.

Application Programming Interface

Introduction

The SLEE provides an Application Programming Interface (API), through which applications may interact with the SLEE and elements within the SLEE, such as applications and interfaces.

All interactions between applications and other SLEE elements are performed via messages based on objects, which are sub-classes of the SleeEvent class.

The SLEE itself only has one type of SleeEvent, the SleeManagementEvent class. Each of the interfaces provided with the SLEE have their own sub-classes of SleeEvent. SleeEvents may be sent as part of a dialog with another SLEE entity, in which case a SleeDialog object is used to associate SleeEvents of the same dialog. Alternatively, messages may be sent as one-off events, which do not require a response or associated message. In this case, there is no SleeDialog object.
SLEE Dialogs

A SleeDialog provides an association between related messages flowing between SLEE entities (applications and interfaces).

Dialogs also store the 'addresses' of the two entities involved in the dialog. Each dialog has an application side and an interface side.

**Note:** When an application instance opens a dialog with another application, the first application instance is considered to be the 'interface' side of the dialog. Also, if a SLEE interface opens a dialog with another SLEE interface, the first SLEE interface is considered to be the 'application' side of the dialog.

Application instances can only open dialogs with interfaces and interfaces can only open dialogs with applications. To enable application to application and interface to interface dialogs, the first application or interface must pretend to be interface or application respectively.

SLEE Events

SLEE events are chunks of shared memory which are used to communicate data within a dialog. Applications and interfaces monitor dialogs for new events.

Management Events

A SLEE management event is an implementation of a SLEE event, which is used to pass management events from the SLEE to applications and interfaces.

The SLEE reserves a number of events user to send management events. These events are added to the first list that has a size greater than, or equal to, 1024 bytes. This means the event count in the check program will show more events than configured in the configuration file.

**Supported management events**

The following SLEE management events are supported.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WATCHDOG</td>
<td>The watchdog monitors the health of all SLEE elements by sending a series of checks to each configured element. If an element fails to respond, the watchdog will take action to restart the process. Initially, it will send the element a SIGABRT. If the process does not die after a period, a SIGKILL will be sent.</td>
</tr>
<tr>
<td>SERVICE_ENABLED</td>
<td>Indicates to an application instance that a service has been enabled. The service handle is passed in the event. Can be used to trigger opening of files, databases, etc.</td>
</tr>
<tr>
<td>SERVICE_DISABLED</td>
<td>Indicates to an application instance that a service has been disabled. The service handle is passed in the event. Can be used to trigger closing of files, databases, etc.</td>
</tr>
<tr>
<td>APPLICATION_END</td>
<td>Indicates that the application is currently being quiesed and that no new calls will be received. An APPLICATION_KILL will be received when all existing calls complete.</td>
</tr>
<tr>
<td>APPLICATION_KILL</td>
<td>Indicates that the application instance should be shutdown and exited. Failure to perform this task within a time period will result in a SIGHTERM being sent to the application instance. If the application instance is still active after a further period, a SIGKILL will be sent.</td>
</tr>
<tr>
<td>Event</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INTERFACE_END</td>
<td>Indicates that the interface is currently being quiesed and that no new calls will be received or should be generated. An INTERFACE_KILL will be received when all existing dialogs are complete.</td>
</tr>
<tr>
<td>INTERFACE_KILL</td>
<td>Indicates that the interface process should be shutdown and exited. Failure to perform this task within a time period will result in a SIGHTERM being sent to the interface process. After a further period, if the interface process is still active, a SIGKILL will be sent.</td>
</tr>
<tr>
<td>DIALOG_CLOSED</td>
<td>Indicates to an interface or application that a dialog has been closed by the other end, but no message data has been sent.</td>
</tr>
<tr>
<td>REREAD_CONFIG</td>
<td>This message is sent to an application or interface to request it to re-read its configuration. This is a user-management event which the SLEE will never transmit on its own. Not all interfaces support this request.</td>
</tr>
<tr>
<td>CALL_INSTANCE_KILL</td>
<td>The specified call instance has been killed.</td>
</tr>
<tr>
<td>REPORT_REQUEST</td>
<td>This message is sent to an application or interface to generate a short report to stdout or a pre-defined file. Not all interfaces support this request.</td>
</tr>
<tr>
<td>CALL_INSTANCE_TIMED_OUT</td>
<td>The specified call instance has timed out.</td>
</tr>
<tr>
<td>DIALOG_TIMED_OUT</td>
<td>The specified dialog has timed out.</td>
</tr>
<tr>
<td>STATUS_REQUEST</td>
<td>A status request is sent to an application or interface to generate a short status summary. The receiver should send back a STATUS_RESPONSE message with the current status of the process. Not all interfaces support this request.</td>
</tr>
<tr>
<td>STATUS_RESPONSE</td>
<td>Contains the response for the STATUS_REQUEST message.</td>
</tr>
</tbody>
</table>
Overview

Introduction

This chapter explains how to configure the Oracle Communications Network Charging and Control (NCC) application.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Configuration Overview</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring the SLEE</td>
<td>10</td>
</tr>
</tbody>
</table>

Configuration Overview

Introduction

This topic provides a high level overview of how the SLEE is configured.

There are configuration options which are added to the configuration files that are not explained in this chapter. These configuration options are required by the application and should not be changed.

Configuration process overview

This table describes the steps involved in configuring the SLEE for the first time.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | The SLEE.cfg file must be configured. This will include configuring:  
      - SLEE maximum values  
      - Watchdog  
      - SLEE interfaces which will be used for this installation (for example, timerIF, cdrIF and/or alarmsIF. |
| 2    | Any SLEE interface or application which has an additional configuration file must be configured. For example, the cdrIF is configured using the cdrIF.cfg file. |

Note: Most installations will require other applications and interfaces to be configured in the SLEE.cfg also. This should be done after the other applications have been installed. For more information about how to configure additional interfaces and applications, see the documentation for the application.
Configuration components

The SLEE is configured by the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Locations</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLEE.cfg</td>
<td>all machines</td>
<td>The only file used to configure the SLEE is <code>SLEE.cfg</code>.</td>
<td>Configuring the SLEE (on page 10).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The configuration file is used to configure the applications, services and interfaces which the SLEE manager will initialize. From this information the SLEE manager also knows how much shared memory to allocate. <code>SLEE.cfg</code> is broken into three sections:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maximum object instances</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Application entries</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interface entries</td>
<td></td>
</tr>
<tr>
<td>Environmental variables</td>
<td>all machines</td>
<td>SLEE supports some environmental variables.</td>
<td>Optional environmental variables (on page 10)</td>
</tr>
<tr>
<td>cdrIF.cfg</td>
<td>all machines</td>
<td>This file configures the cdrIF.</td>
<td></td>
</tr>
<tr>
<td>tcRelayMappings.def</td>
<td>all machines</td>
<td>This file configures the tcRelayApp.</td>
<td>Configuration (on page 30)</td>
</tr>
</tbody>
</table>

Configuring the SLEE

Introduction

The SLEE must be configured at start-up. The default configuration file is `/IN/service_packages/SLEE/etc/SLEE.cfg`. You can change this by setting the SLEE_FILE environmental variable.

Optional environmental variables

SLEE supports the following environmental variables (usually declared in acs_oper's SLEE_FILE_:

SLEE_FILE

Syntax: Description: The location of the file which stores the shared memory keys for the SLEE's shared memory.

Type: Optionality: Optional.

Allowed: Default: `/tmp/slee`

Notes: All SLEE processes must use the same SLEE_FILE.

Example:
Modifying SLEE.cf

Each section of the SLEE.cfg configuration file is detailed below, with examples of appropriate settings. The SLEE must be restarted for any configuration changes to take effect. For more information on restarting the SLEE, see Tools and Utilities (on page 23).

Maximum values

The following maximum lengths apply to names used in the SLEE.cfg file:

- 20 characters for interface names
- 20 characters for service names
- 40 characters for application names

The first section of SLEE.cfg contains the maximum number of each type of object which can be held in shared memory. If this value is exceeded, an exception will be thrown and an entry made in the syslog. In many cases this will cause the SLEE and all processes to restart.

Format:

- MAXAPPLICATIONS=max
- MAXSERVICES=max
- MAXSERVICEHANDLES=max
- MAXSERVICEKEYS=max
- MAXDIALOGS=max
- MAXEVENTS=max [size]
- MAXCALLS=max
- MAXINTERFACES=max
- MAXEVENTTYPES=max

The available parameters are:

MAXAPPLICATIONS

- Syntax: MAXAPPLICATIONS=max
- Description: The maximum number of application objects which can be held in shared memory.
- Type: Integer
- Optionality: Mandatory
- Allowed: none
- Default: none
- Notes: This also sets the maximum number of APPLICATION (on page 19) entries which can be active in the SLEE.cfg file.
- Example: MAXAPPLICATIONS=5

MAXSERVICES

- Syntax: MAXSERVICES=max
- Description: The maximum number of service objects which can be held in shared memory.
- Type: Integer
- Optionality: Mandatory
- Allowed: none
- Default: none
- Notes: This also sets the maximum number of SERVICE entries (on page 17) which can be active in the SLEE.cfg file.
- Example: MAXSERVICES=5
MAXSERVICEHANDLES

**Syntax:**

```
MAXSERVICEHANDLES=max
```

**Description:**
The maximum number of service handles.

**Type:**
Integer

**Optionality:**
Optional (default used if not set).

**Allowed:**
10

**Notes:**
MAXSERVICEHANDLES has to be greater than or equal to the number of distinct service handles specified in `SERVICE entries` on page 17 in `SLEE.cfg`

**Example:**
MAXSERVICEHANDLES=20

MAXSERVICEKEYS

**Syntax:**

```
MAXSERVICEKEYS=max
```

**Description:**
The maximum number of service key objects which can be held in shared memory.

**Type:**
Integer

**Optionality:**
Optional (default used if not set).

**Allowed:**
If this parameter is not specified, the count of serviceKey entries in the configuration files are used.

**Notes:**
This also sets the maximum number of `SERVICEKEY` (on page 18) entries which can be active in the `SLEE.cfg` file.

**Example:**
MAXSERVICEKEYS=5

MAXDIALOGS

**Syntax:**

```
MAXDIALOGS=max
```

**Description:**
The maximum number of dialog objects which can be held in shared memory.

**Type:**
Integer

**Optionality:**
Mandatory

**Allowed:**
none

**Notes:**
MAXEVENTS can be specified more than once for multiple lists of differing sizes and the effects are cumulative.

This list will then have its count modified (when the SLEE starts up) to handle the management event reservation. This affects the check tool event count. For more information, see `check` (on page 25).

If the MAXEVENTS values are exceeded when the system is running, no more...
events or calls will be accepted and alarm messages will be sent.

**Example:**
To get 500 events of size 2k, and 200 of size 4k, set:

```
MAXEVENTS=500 2048
MAXEVENTS=200 4096
```

**size**

**Syntax:**
```
size
```

**Description:**
`MAXEVENTS` (on page 12) supports the `size` parameter which allows you to set the maximum data segment size of an event.

**Type:**
Integer

**Optionality:**
Optional (default used if not set).

**Allowed:**
1024

**Default:**
1024

**Notes:**
There must be at least one list with a size equal to or greater than 1024 bytes.

**Example:**
For an example of this parameter in context, see `MAXEVENTS` (on page 12).

**MAXCALLS**

**Syntax:**
```
MAXCALLS=max
```

**Description:**
The maximum number of call objects which can be held in shared memory.

**Type:**
Integer

**Optionality:**
Mandatory

**Allowed:**

**Default:**
one

**Notes:**
If the MAXCALLS values are exceeded when the system is running, no more events or calls will be accepted and alarm messages will be sent.

**Example:**
MAXCALLS=500

**MAXINTERFACES**

**Syntax:**
```
MAXINTERFACES=max
```

**Description:**
The maximum number of interface objects which can be held in shared memory.

**Type:**
Integer

**Optionality:**
Mandatory

**Allowed:**

**Default:**

**Notes:**
This also sets the maximum number of INTERFACE entries which can be active in the `SLEE.cfg` file.

**Example:**
MAXINTERFACES=5

**MAXEVENTTYPES**

**Syntax:**
```
MAXEVENTTYPES=max
```

**Description:**
The maximum number of Event Type objects which can be held in shared memory.

**Type:**
Integer

**Optionality:**
Mandatory

**Allowed:**

**Default:**


Notes:

Example: MAXEVENTTYPES=5

Other parameters
These parameters are set after the Maximum values configuration in SLEE.cfg.

NOWIPESOCKETS
Syntax: NOWIPESOCKETS=\textit{int}
Description: How to handle pre-existing sockets on startup.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 0 Socket files corresponding to SLEE.cfg-specified interfaces are removed.
          anything else No socket files will be removed.
Default: 0
Notes: This setting is designed for use in testing environments where more than one SLEE is running concurrently. It should not be used in production.

Example:

INTERFACE
You use the INTERFACE entries to configure the interfaces you want the SLEE to run. You can run multiple instances of the same interface. You specify the number of instances of a particular interface to run in an INTERFACE entry parameter. Messages can be sent directly to an interface handle, or to a service key.

Installing and removing packages may add or remove interface entries. Do not manually remove entries from the SLEE.cfg file that have been added by the package installation or removal process.

Usage:
\texttt{INTERFACE=interface\_handle interface\_name interface\_path instance\_count interface\_type [int\_event\_count dialog\_count]}

INTERFACE parameters

\texttt{interface\_handle}
Syntax: To see this parameter in context, see Usage (on page 14).
Description: The unique name that identifies this SLEE interface. You can run multiple instances of a particular interface by specifying the number of instances to run in the instance\_count (on page 15) parameter.
Type: String
Optionality: Required.
Allowed: The last character in the name should not be numeric.
Default: None
Notes: If you configure the SLEE to run multiple instances of an interface, the SLEE appends a unique number to each instance of the interface handle, at startup. The system also creates a separate log file for each interface instance and appends the number of the interface instance to the log file name.
Example: For an example of this parameter used in context, see Example INTERFACE Configuration (on page 16).
interface_name
Syntax: To see this parameter in context, see Usage (on page 14).
Description: The name of the executable file that enables this interface.
Type: String
Optionality: Required
Allowed: A valid NCC binary.
Default: None
Notes:
Example: For an example of this parameter used in context, see Example INTERFACE Configuration (on page 16).

interface_path
Syntax: To see this parameter in context, see Usage (on page 14).
Description: The full path to the interface binary file.
Type: String
Optionality: Required
Allowed: A valid path.
Default: None
Notes:
Example: For an example of this parameter used in context, see Example INTERFACE Configuration (on page 16).

instance_count
Syntax: To see this parameter in context, see Usage (on page 14).
Description: The number of instances to run of the defined SLEE interface.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: A numeric value that is greater than or equal to 1(one).
Default: 1
Notes: If you configure to run more than one instance of a particular interface, then the SLEE creates a unique ID for each instance of the interface by appending a number to the interface name. The number will be in the range 0 to n-1, where n is the configured instance_count value. A number will not be appended to the interface name if there is only one instance of the interface.
Example: For an example of the parameter used in context, see Example INTERFACE Configuration (on page 16).

interface_type
Syntax: To see this parameter in context, see Usage (on page 14).
Description: The type of interface.
Type: String
Optionality: Required
Allowed: EVENT
           UGD
Default: None
Notes:
Example: For an example of this parameter used in context, see Example INTERFACE Configuration (on page 16).

**int_event_count**

Syntax: To see this parameter in context, see Usage (on page 14).

Description: The maximum number of events allowed to wait for processing before call limiting for the interface starts.

Type: Integer

Optionality: Optional, but required if dialog_count is set.

Allowed: Unlimited

Notes: If the number of events exceeds the limit specified in the configuration file, then any further attempts to create another dialog to the interface will fail. This failure will then be handled by the process attempting to create the dialog.

Example: For an example of this parameter used in context, see Example INTERFACE Configuration (on page 16).

**dialog_count**

Syntax: To see this parameter in context, see Usage (on page 14).

Description: The maximum number of dialogs that can be open on the interface.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Unlimited

Notes: The SLEE tracks the number of dialogs open on the interface. If the interface has exceeded the limit specified in the configuration file, any further attempts to create a dialog on the interface will fail.

Example: For an example of this parameter used in context, see Example INTERFACE Configuration (on page 16).

**Example INTERFACE Configuration**

These lines in SLEE.cfg configure three timer interface instances, one replication interface instance, and two notificationIF interface instances:

```
INTERFACE=Timer timerIF /IN/service_packages/SLEE/bin 3 EVENT
INTERFACE=Replication replicationIF.sh /IN/service_packages/SLEE/bin 1 EVENT
INTERFACE=notificationIF notificationIF /IN/service_packages/SLEE/bin 2 UDG
```

At startup, the SLEE creates a unique ID for each instance of an interface by appending a number in the range 0 to n-1 to the interface name, where n is the number of interface instances configured in the instance_count (on page 15) parameter. If there is only one instance of an interface, then the SLEE does not append a number to the interface name. This means that the three Timer interface instances in the example would have the following IDs: Timer0, Timer1, and Timer2. Whereas the ID for the single instance of the Replication interface would be: Replication.

**watchdog**

This section defines the location and cycle time for the watchdog. You should not need to alter these settings.

```
WATCHDOG=/IN/service_packages/SLEE/bin/ watchdog
WATCHDOGCYCLETIME=30
```

For more information, see watchdog (on page 31).
SERVICE entries

The SERVICE entries define each service object to be created in shared memory. The service name, priority and the name of the application that provides this service are defined here. Each service must be associated with an application.

Note: You cannot have more SERVICES than the number allowed by MAXSERVICEHANDLES (on page 12).

Usage:

SERVICE=serviceName priority appName serviceHandle [callCount]

The available parameters are:

serviceName

Syntax: To see this parameter in context, see SERVICE entries (on page 17).
Description: The name of the service provided by an application.
Type: String
Optionality: Mandatory
Allowed: 
Default: 
Notes: This matches the dest (on page 19) parameter in the SERVICEKEY (on page 18) entry for the service keys which will use this service. It must match at least one service key entry.
Example: For an example of this parameter used in context, see Examples (on page 18).

priority

Syntax: To see this parameter in context, see SERVICE entries (on page 17).
Description: The priority the scheduler gives this service.
Type: Integer
Optionality: Mandatory
Allowed: 
Default: 
Notes: 
Example: For an example of this parameter used in context, see Examples (on page 18).

appName

Syntax: To see this parameter in context, see SERVICE entries (on page 17).
Description: The name of the application which enables this service.
Type: String
Optionality: Mandatory
Allowed: 
Default: 
Notes: This matches to the appName (on page 19) parameter in the APPLICATION (on page 19) for the application which will handle this service. It must match at least one application entry.
Example: For an example of this parameter used in context, see Examples (on page 18).


**serviceHandle**

**Syntax:**
To see this parameter in context, see *SERVICE entries* (on page 17).

**Description:**
The service handle which is sent to the application to enable it to provide more than one service.

**Type:**
String

**Optionality:**
Mandatory

**Allowed:**

**Default:**

**Notes:**
It must match the service handle defined in the application entry this service entry links to.

**Example:**
Service handles for slee_acs must match serviceNames in ServiceEntries in *acs.conf*. For more information about ACS ServiceEntries, see *ACS Technical Guide*.

There will typically be multiple lines of this type for each appName as one application will usually handle more than one service.

**Example:**
For an example of this parameter used in context, see *Examples* (on page 18).

**callCount**

**Syntax:**
To see this parameter in context, see *SERVICE entries* (on page 17).

**Description:**
The maximum number of concurrent calls which can be processed by this service.

**Type:**
Integer

**Optionality:**
Optional

**Allowed:**

**Default:**
unlimited

**Notes:**
If the number of calls active on this service exceeds the specified limit, all attempts to create a call for this service will fail.

**Example:**
For an example of this parameter used in context, see *Examples* (on page 18).

**Examples**

This text shows some examples of SERVICE entries in a *SLEE.cfg* file.

```
SERVICE=PREPAID 1 slee_acs CCS
SERVICE=ACS_Outgoing 1 slee_acs ACS_Outgoing
```

**SERVICEKEY**

The service key entries define each service key. They also include information on which service or interface will handle this service key. Each service key must be associated with either a service or an interface instance.

Service keys have the following configuration options:

```
SERVICEKEY=keyType serviceKey dest
```

The available parameters are:

**keyType**

**Syntax:**
To see this parameter in context, see *SERVICEKEY* (on page 18).

**Description:**
The type of service key.

**Type:**
Integer

**Optionality:**

**Allowed:**

serviceKey
Syntax: To see this parameter in context, see SERVICEKEY (on page 18).
Description: The service key from interface.
Type: Optionality:
Allowed: Format depends on interface.
Default:
Notes: Example: For an example of this parameter used in context, see Examples (on page 19).

dest
Syntax: To see this parameter in context, see SERVICEKEY (on page 18).
Description: Service name or interface name.
Type: Optionality:
Allowed: Must match a serviceName (on page 17) or ifHandle.
Default:
Notes: Example: For an example of this parameter used in context, see Examples (on page 19).

Examples
This text shows examples of SERVICEKEY entries.

SERVICEKEY=INTEGER 101 PREPAID
SERVICEKEY=INTEGER 1 0800

The serviceKey depicts the service key that this application will handle. There will typically be multiple lines of this type for each appName as one application will usually handle more than one service key.

APPLICATION
The application entry enables the SLE to execute the binary files.

Usage:
APPLICATION=appName execName execDir startInstances maxInstances [appEventCount]
The available parameters are:

appName
Syntax: To see this parameter in context, see APPLICATION (on page 19).
Description: The name of the application.
Type: Optionality:
Allowed: Mandatory
Default: none
Notes: This is used to refer to the application in other parts of the configuration file,
including SERVICE entries (on page 17), where it must be matched by appName (on page 17).

Example: For an example of this parameter used in context, see Example (on page 21).

execName
Syntax: To see this parameter in context, see APPLICATION (on page 19).
Description: The name of the binary file to be run.
Type: String
Optionality: Mandatory
Allowed:
Default: none
Notes:
Example: For an example of this parameter used in context, see Example (on page 21).

eexecDir
Syntax: To see this parameter in context, see APPLICATION (on page 19).
Description: Full path to the directory where the executable binary is stored.
Type: String
Optionality: Optional (default used if not set).
Allowed: Any directory path.
Default: "/IN/service_packages/SLEE/bin"
Notes:
Example: For an example of this parameter used in context, see Example (on page 21).

startInstances
Syntax: To see this parameter in context, see APPLICATION (on page 19).
Description: The number of instances of the application the SLEE should initially start.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 1
Notes:
Example: For an example of this parameter used in context, see Example (on page 21).

maxInstances
Syntax: To see this parameter in context, see APPLICATION (on page 19).
Description: The maximum number of instances of the application that the SLEE will support.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 1
Notes:
Example: For an example of this parameter used in context, see Example (on page 21).
appEventCount

Syntax: To see this parameter in context, see APPLICATION (on page 19).

Description: The maximum number of events allowed to be awaiting processing before call limiting for the application starts.

Type: Integer

Optionality: Optional

Allowed:

Default:

Notes:

- app event count applies to the application as a whole, that is, all the application instances combined. If app event count is set to 1000 and start num instance and max num instance are both set to 2, then two application instance processes will run and each one can have up to 500 events queued.

- If the number of events exceeds the limit specified in the configuration file, then any further attempts to create another call instance will fail.

Example: For an example of this parameter used in context, see Example (on page 21).

Example

This text shows an example of an APPLICATION entry.

APPLICATION=appExample appExample ../appExample 1 1

Example SLEE.cfg file

This is an example of the configuration part of a SLEE.cfg file.

MAXAPPLICATIONS=10
MAXSERVICES=10
MAXSERVICEHANDLES=10
MAXSERVICEKEYS=20
MAXDIALOGS=70000
MAXEVENTS=50000
MAXCALLS=25000
MAXINTERFACES=20
MAXEVENTTYPES=30
MAXCORRELATIONIDS=10000

INTERFACE=Timer timerIF /IN/service_packages/SLEE/bin 1 EVENT
INTERFACE=acsStatsLocalSLEE acsStatsLocalSLEE /IN/service_packages/ACS/bin 1 EVENT
INTERFACE=Replication replicationIF.sh /IN/service_packages/SLEE/bin 1 EVENT
INTERFACE=hssScIf hssScIf.sh /IN/service_packages/SLEE/bin 1 EVENT

WATCHDOG=/IN/service_packages/SLEE/bin/ watchdog
WATCHDOGCYCLETIME=30

# SLEE Process Manager (statistics collection)
#INTERFACE=sleeProcMan sleeProcMan /IN/service_packages/SLEE/bin 1 UDG

# APPLICATION
APPLICATION=mngApp mngApp /IN/service_packages/SLEE/bin 1 1

# SERVICE
SERVICE=ACS 1 slee_acs ACS
SERVICE=ACS_Outgoing 1 slee_acs ACS_Outgoing

# SERVICEKEY
SERVICEKEY=INTEGER 111 ACS
SERVICEKEY=INTEGER 110 ACS_Outgoing
Overview

Introduction

This chapter explains how to use the utilities provided with the SLEE. To:

- Start the SLEE, see `stop.sh` (on page 24).
- Shut down the SLEE, see `stop.sh` (on page 24).
- Remove Shared Memory and semaphores, see `clean` (on page 24).
- Display what SLEE resources are in use, see `check` (on page 25).
- Create a statistics.bin file for statsIF, see `sfVerify` (on page 27).

Warning: All these scripts must be run from `/IN/service_packages/SLEE/bin`. Unpredictable results will occur if run from elsewhere.

In this chapter

This chapter contains the following topics.

- `slee.sh` 23
- `stop.sh` 24
- `clean` 24
- `check` 25
- `sfVerify` 27

`slee.sh`

Purpose

`slee.sh` provides a standardized way of starting the SLEE.

Warning: Running this script while the SLEE is already running will result in the SLEE becoming unstable.

Configuration

The `slee.sh` does not support any configuration options.

Failure

If `slee.sh` fails, the SLEE will not start properly. To ensure you start the SLEE in a stable environment, complete the following before you run `slee.sh` again:

- Run `stop.sh`
- Run `clean`
- Ensure all SLEE processes are killed
Output
slee.sh writes error messages to the system messages file.

stop.sh

Purpose
The stop.sh script shuts down the SLEE in a controlled manner. This ensures the SLEE shared memory and semaphores are cleared.

Configuration
stop.sh does not support any configuration options.

Failure
If the stop.sh script has failed, the SLEE may not have been shut down properly. Attempt to run the stop.sh script again, and run clean to ensure all SLEE shared memory has been properly removed and all processes have been removed.

Output
stop.sh writes error messages to the system messages file.

clean

Purpose
The clean tool uses the Unix clean tool to remove an current SLEE shared memory and semaphores. This must be completed if the SLEE has exited without being shut down properly, for example if there was a network outage.

Startup
clean is started by acs_oper from the command line using the following command:
/IN/service_packages/SLEE/bin/clean

Configuration
clean does not support any configuration options.

Failure
If clean fails, SLEE Shared Memory may still exist. Attempt to rerun the script.

Output
clean writes error messages to the system messages file.
check

Purpose

check provides a method of monitoring the SLEE. It can produce either periodic reports or general reports.

Events

The number of events (total) reported by the check tool will not match the total event number specified in the configuration file. This is because SLEE reserves a set of events for exclusive use by the watchdog process. These events are used to clean up the SLEE if a runaway process allocates all the available SLEE resources and needs to be cleaned up. The additional number of events is calculated as:

\[ \text{Extra Event} = \text{Max Dialogs} \times 2 + \text{Max Application Instances} \times 2 + \text{Max Interfaces} \times 2 \]

These events are added into the list with a size greater (or equal) to the default of 1024.

Configuration

check supports the following command-line options:

Usage:

```
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-v</td>
<td></td>
<td>Verbose, Include data in event dumps.</td>
</tr>
<tr>
<td>-c</td>
<td></td>
<td>Include floating calls in the report.</td>
</tr>
<tr>
<td>-C</td>
<td></td>
<td>Display all calls in the report.</td>
</tr>
<tr>
<td>-d</td>
<td></td>
<td>Display active dialogs in the report.</td>
</tr>
<tr>
<td>-D</td>
<td></td>
<td>Display all dialogs in the report.</td>
</tr>
<tr>
<td>-e</td>
<td></td>
<td>Display floating events in the report.</td>
</tr>
<tr>
<td>-E</td>
<td></td>
<td>Display all events in the report.</td>
</tr>
<tr>
<td>-f</td>
<td></td>
<td>Display free object counts in the report.</td>
</tr>
<tr>
<td>-q</td>
<td></td>
<td>Display event queues.</td>
</tr>
<tr>
<td>-b</td>
<td></td>
<td>Batch mode. Only print the report header once.</td>
</tr>
<tr>
<td>-n</td>
<td></td>
<td>Display top ( N_{\text{items}} ) items.</td>
</tr>
</tbody>
</table>

\( N_{\text{items}} \) The number of items to display when specifying the \(-n\) parameter

delay The number of seconds before the specified report is repeated.

count 1 The number of times the report should be repeated.

Note: To access the check script’s reports menu (and gain access to the general reports), run the script without the seconds variable.
Main Menu options

To gain access to the check script's report menu, run the command line invocation without any options.

Example: /IN/service_packages/SLEE/bin/check

Main menu options: Once you have started the check script, you will see a menu with 11 options as shown below.

Check which type of object?

1: Dialogs
2: Events
3: Calls
4: Services
5: Applications
6: Application Instances
7: Interface Instances
8: General Status
9: Free Objects
a: Event Queues
q: Quit

Select:
The options are executed by typing the character before the colon. This table describes the main menu options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Dialogs</td>
<td>Reports the number of open SLEE dialogs.</td>
</tr>
<tr>
<td>2: Events</td>
<td>Reports the number of current SLEE events.</td>
</tr>
<tr>
<td>3: Calls</td>
<td>Reports the number of current calls.</td>
</tr>
<tr>
<td>4: Services</td>
<td>Reports the number of SLEE services which are running.</td>
</tr>
<tr>
<td>5: Applications</td>
<td>Reports the number of SLEE applications which are currently running.</td>
</tr>
<tr>
<td>6: Application Instances</td>
<td>Reports the number of instances of SLEE applications which are currently running.</td>
</tr>
<tr>
<td>7: Interface Instances</td>
<td>Reports the number of SLEE interface instances which are currently running.</td>
</tr>
<tr>
<td>8: General Status</td>
<td>Reports general information and statistics about the SLEE.</td>
</tr>
<tr>
<td>9: Free Objects</td>
<td>Reports the number of free objects still available in the SLEE shared memory.</td>
</tr>
<tr>
<td>a: Event Queues</td>
<td>Reports the number of events queued and waiting for the application or interface to process.</td>
</tr>
<tr>
<td>q: Quit</td>
<td>Exits from check.</td>
</tr>
</tbody>
</table>

Output

check writes reports to stdout. The different reports have different formats. Reports will be the same whether run from the command line or the main menu options.

Note: The number of events (total) reported by the check tool will not match the total event number specified in the configuration file. This is because SLEE reserves a set of events for exclusive use by the watchdog process.

check writes error messages to the system messages file.
General reports

If you choose option 8 from the menu, the check script will provide you with a summary report of the information available from the other reports, and return you to the menu.

Example:  This is an example of a general report.

Select:  8
SLEE Status Report

Service:   ACS
Using application:   0xc0013d28

Service:   ACS_Outgoing
Using application:   0xc0013d28

Application:   slee_acs at 0xc0013d28
Contains the following Instances....
Instance at :   0xc0014b88
Process ID:   5493
Status:   3
Call Count:   0

Free Dialogs:   70000
Free Applications:   9
Free Application Instances:   90
Free Services:   8
Free Events:   49998
Free Calls:   25000

Exiting check

To exit from the check report running in periodic report mode, press Ctrl+C.
To exit from the menu choose option q.

sfVerify

Purpose

sfVerify creates the statistics.bin file that is needed to run statsIF.

Configuration

sfVerify supports the following command-line options:

Usage:

sfVerify [-v--verbose-c-commit-f-force] [-d path --dir path] [-o filename --output filename] [-s KB --size KB]

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-v</td>
<td>false</td>
<td>Controls the amount of information output from the program.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-c --commit</td>
<td>false</td>
<td>Commits the changes to the stats interface. This will only work if the SLEE is running and the stats interface is configured. This parameter writes the output file and then signals the stats interface to reread the configuration file statistics.bin.</td>
</tr>
<tr>
<td>-f --force</td>
<td>false</td>
<td>Force the commit without asking the user.</td>
</tr>
<tr>
<td>-d --dir</td>
<td>stats_defn</td>
<td>The directory to import the statistics definitions from.</td>
</tr>
<tr>
<td>-o --output</td>
<td>output.defn</td>
<td>The name of the output file.</td>
</tr>
<tr>
<td>-s --size</td>
<td>128</td>
<td>The maximum size in Kb for the stats interface output files.</td>
</tr>
</tbody>
</table>

**Note:** Any of the parameters (except the period) can either be a single word, or specified as a quoted-delimited string.

**Import files**

The import files for the statistics interface take the following form:

```
applicationName statisticName description period comment
```

**Output**

The output file used by the statistics interface is:

```
/IN/service_packages/SLEE/etc/statistics.bin
```

sfVerify writes error messages to the system messages file.
Overview

Introduction

This chapter explains the processes which run automatically as part of the application. These processes are started automatically by one of the following:

- inittab
- crontab
- Service Logic Execution Environment SLEE

Note: This chapter also includes some plug-ins to background processes which do not run independently.

In this chapter

This chapter contains the following topics.

replicationIF 29
tcRelayApp 30
watchdog 31

replicationIF

Purpose

replicationIF responds to SLEE replication events by sending data to another machine (usually the SMS).

Startup

replicationIF is started by the following line in SLEE.cfg:

\[
\text{INTERFACE=}\text{replicationIF} \text{ replicationIF.sh } /IN/service\_packages/SLEE/bin 1 \text{ EVENT}
\]

For more information about using the INTERFACE entry, see INTERFACE (on page 14).

Configuration

replicationIF supports the following command line parameters:

\[
\text{replicationIF } -r \text{ node } -d \text{ microsecs}
\]

- \( r \)

Syntax: \(-r \text{ node}\)

Description: The node number of the requester node (that is, the node number of the replicationIF itself).

Type: Integer

Optionality: Mandatory (disallowed default of 0 used if not set).
Allowed: A number between 512 and 1023.
Default: 0
Notes: 
Example: -r 601

-d
Syntax: -d microsecs
Description: The number of microseconds between processing large SLEE events.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 1000
Notes: 
Example: 

## tcRelayApp

### Purpose

tcRelayApp is a SLEE application which relays TCAP primitives. This enables SLEE interfaces to send TCAP primitives to other SLEE interfaces (particularly TCAP IF).

You can use tcRelayApp to add destination number and originating number information to the TCAP primitive.

### Startup

tcRelayApp is started by the following line in SLEE.cfg:

```
APPLICATION=tcRelayApp tcRelayApp.sh /IN/service_packages/SLEE/bin 1 1
```

### Configuration

tcRelayApp supports the following parameters in each line of tcRelayMappings.def:

```
serviceHandle IF name dest ssn dest_pc dest_GT orig_ssn orig_pc
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceHandle</td>
<td></td>
<td>Incoming service handler. The SLEE service handle this message has been sent to. (Required.)</td>
</tr>
<tr>
<td>IF name</td>
<td></td>
<td>Outgoing interface name. The SLEE interface name the message should be forwarded to. (Required.)</td>
</tr>
<tr>
<td>dest ssn</td>
<td></td>
<td>Destination SSN. The Destination SSN value which the TCAP primitive should have. (Required.)</td>
</tr>
<tr>
<td>dest pc</td>
<td></td>
<td>Destination PC. The Destination Point Code value which the TCAP primitive should have. (Required.)</td>
</tr>
<tr>
<td>dest gt</td>
<td></td>
<td>Destination GT. The Destination Global Title value which the TCAP primitive should have. (Optional, use &quot;-&quot; to not specify.)</td>
</tr>
<tr>
<td>orig ssn</td>
<td></td>
<td>Originating SSN. The Originating SSN value which the</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| orig pc   |         | TCAP primitive should have. (Optional.)  
|           |         | Originating PC. The Originating PC value which the TCAP primitive should have. (Optional.) |

### Failure

If tcRelayApp fails, TCAP primitives which must be relayed through the SLEE will fail. If the process is not running, the SLEE watchdog will attempt to restart the service.

### Output

tcRelayApp writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SLEE/tmp/tcRelayApp.log

### watchdog

#### Purpose

The watchdog is responsible for the maintenance of the processes and ensuring the current time of day is correct in the shared memory segment.

If it detects that either some application (or interface) has stopped processing events it shuts it down and cleans up any dialogs associated with it.

Once everything is clean, it then restarts the failed process.

The watchdog has three timers, which run concurrently:

1. Every 10 milliseconds the current time of day is written to the shared memory segment. If this is unsuccessful, the error code is inserted in the shared memory so that the other SLEE processes are able to determine an error has occurred and respond appropriately.

2. Every 100 milliseconds the watchdog scans all the SLEE processes looking at their total event count. If this number has changed, due to the process handling events, the process is considered VALID and no other action is taken. If the process has not handled any events it is marked as SUSPECT and a WATCHDOG management event is sent to the process.

3. Any SUSPECT processes are checked every WATCHDOG_CYCLE_TIME (defaults to 30 seconds) to see if they have processed any events since the watchdog last checked. If they have, the process is added back on to the VALID list and handled normally. Otherwise the process is aborted and restarted.

Another responsibility for the watchdog is keeping track of SLEE resource usage. If the resource drops below 80% of the start value, a warning is raised alerting the user to the error. A notice is posted when the value rises back up to 70% of the start value. The watchdog keeps track of the resource usage for the dialog, call instance and event lists.

When the watchdog begins a check loop, it starts a timer running to ensure that it will not remain deadlocked on a semaphore forever. If the timer expires the watchdog will restart the SLEE.

#### Startup

watchdog is started by the following line in SLEE.cfg:

```
APPLICATION=tcRelayApp tcRelayApp.sh /IN/service_packages/SLEE/bin 1 1
```
Configuration

watchdog accepts the following configuration options from the SLEE.cfg file. These are set at installation and should not need changing.

This section defines the location and cycle time for the watchdog. You should not need to alter these settings.

WATCHDOG=/IN/service_packages/SLEE/bin/ watchdog
WATCHDOGCYCLETIME=30

The available parameters are:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WATCHDOG</td>
<td></td>
<td>The path and binary filename for the watchdog executable.</td>
</tr>
<tr>
<td>WATCHDOGCYCLETIME</td>
<td>30</td>
<td>The number of seconds between checks on SUSPECT processes. For more information about SUSPECT processes, see Purpose (on page 31).</td>
</tr>
</tbody>
</table>

Failure

If the watchdog fails, SLEE processes will not be monitored. This will mean SLEE processes are not restarted if they fail. You must restart the SLEE. For more information about restarting the SLEE, see Tools and Utilities (on page 23).

Output

watchdog writes error messages to the system messages file.
Overview

Introduction

This chapter explains the important processes on each of the server components in the NCC, and a number of example troubleshooting methods which will help aid the troubleshooting process before raising a support ticket.

In this chapter

This chapter contains the following topics.

Common Troubleshooting Procedures
Possible Problems

Common Troubleshooting Procedures

Introduction

Refer to *NCC System Administrator's Guide* for troubleshooting procedures common to all NCC components.

Checking current processes

You can check which processes are running using the standard UNIX command: ps. To find processes being run by Oracle software, you can grep for the string 'oper', which will display all processes being run by the application operator accounts (for example, acs_oper, ccs_oper and smf_oper).

**Note:** Some processes which are required for proper functioning may be run by other users, including root or the user which runs the webserver.

**Example command:** ps -ef | grep oper

For more information about the ps command, see the system documentation for the ps command.

You can also check how much of the processor a process is using by running the standard UNIX tool: top. If you have some baseline measurements, you will be able to compare it with the current load.

**Example command:** top

**Tip:** Some processes should only have one instance. If there are two or more instances, this may indicate a problem. For example, there will usually only be one timerIF running on each SLC.

For more information about which processes should be running on each node, check the Process List for each node in *Installation*. 
Checking configuration files

One of the significant areas where faults can occur and be remedied is in the configuration of processes. Configuration files can be edited by any standard text editor. A backup of the existing configuration file should always be taken before editing a configuration file.

For more information about the configuration files used in this application, see Configuration.

For more information about the configuration file for a specific program or tool, see the section named after the binary in question.

Possible Problems

Introduction

This topic lists common problems and actions which can be taken to investigate or solve them. This list enables you to check for alarms based on the overall behavior you are experiencing.

SLEE failing on startup

This table describes possible reasons why the SLEE may be failing to startup:

<table>
<thead>
<tr>
<th>Reason</th>
<th>Remedy</th>
<th>Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>sleeStartup could not parse the SLEE.cfg file.</td>
<td>Check that the SLEE.cfg file exists in the expected location and that it can be read.</td>
<td>For more information about the SLEE.cfg file, see Configuration (on page 9).</td>
</tr>
</tbody>
</table>
Overview

Introduction
This chapter provides details of the installation and removal process for the application.

In this chapter

This chapter contains the following topics.
Installation and Removal Overview 35
Checking the Installation 35

Installation and Removal Overview

Introduction
For information about the following requirements and tasks, see *NCC Installation Guide*:

- NCC system requirements
- Pre-installation tasks
- Installing and removing NCC packages

SLEE packages
An installation of SLEE includes the following packages, on the:

- SLC:
  - SLEE
- VWS:
  - SLEE

Checking the Installation

Introduction
Refer to this checklist to ensure that SLEE has installed correctly.

Checklist
Follow these steps in this checklist to ensure the SLEE has been installed on the SLC machine correctly.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log onto the machine as root.</td>
</tr>
<tr>
<td>Step</td>
<td>Action</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 2    | Check the following directory structure exists with subdirectories:  
      |   ● /IN/service_packages/SLEE |
| 3    | Check the directories and subdirectories are all owned by:  
      | root user (group other) |
NCC Glossary of Terms

ACS
Advanced Control Services configuration platform.

API
Application Programming Interface

C7
See SS7.

CC
Country Code. Prefix identifying the country for a numeric international address.

CCS
1) Charging Control Services (or Prepaid Charging) component.
2) Common Channel Signalling. A signalling system used in telephone networks that separates signalling information from user data.

CDR
Call Data Record

Note: The industry standard for CDR is EDR (Event Detail Record). Over time EDR will replace CDR in the Oracle documentation.

Connection
Transport level link between two peers, providing for multiple sessions.

cron
Unix utility for scheduling tasks.

crontab
File used by cron.

GPRS
General Packet Radio Service - employed to connect mobile cellular users to PDN (Public Data Network- for example the Internet).

GT
Global Title.
The GT may be defined in any of the following formats:
- Type 1: String in the form "1,<noa>,<BCD address digits>"
- Type 2: String in the form "2,<trans type><BCD address digits>"
• Type 3: String in the form "3,<trans type>,<num plan>,<BCD address digits>"
• Type 4: String in the form "4,<trans type>,<num plan>,<noa>,<BCD address digits>"

The contents of the Global Title are defined in the Q713 specification, please refer to section 3.4.2.3 for further details on defining Global Title.

**HLR**

The Home Location Register is a database within the HPLMN (Home Public Land Mobile Network). It provides routing information for MT calls and SMS. It is also responsible for the maintenance of user subscription information. This is distributed to the relevant VLR, or SGSN (Serving GPRS Support Node) through the attach process and mobility management procedures such as Location Area and Routing Area updates.

**HPLMN**

Home PLMN

**HTML**

HyperText Markup Language, a small application of SGML used on the World Wide Web.

It defines a very simple class of report-style documents, with section headings, paragraphs, lists, tables, and illustrations, with a few informational and presentational items, and some hypertext and multimedia.

**IN**

Intelligent Network

**INAP**

Intelligent Network Application Part - a protocol offering real time communication between IN elements.

**ISDN**

Integrated Services Digital Network - set of protocols for connecting ISDN stations.

**ISUP**

ISDN User Part - part of the SS7 protocol layer and used in the setting up, management, and release of trunks that carry voice and data between calling and called parties.

**LNP**

Local Number Portability

**Messaging Manager**

The Messaging Manager service and the Short Message Service components of Oracle Communications Network Charging and Control product. Component acronym is MM (formerly MMX).

**MM**

Messaging Manager. Formerly MMX, see also XMS (on page 41) and Messaging Manager (on page 38).
**MSISDN**
Mobile Station ISDN number. Uniquely defines the mobile station as an ISDN terminal. It consists of three parts; the country code (CC), the national destination code (NDC) and the subscriber number (SN).

**MT**
Mobile Terminated

**MTP**
Message Transfer Part (part of the SS7 protocol stack).

**Oracle**
Oracle Corporation

**PC**
Point Code. The Point Code is the address of a switching point.

**PLMN**
Public Land Mobile Network

**SCCP**
Signalling Connection Control Part (part of the SS7 protocol stack).

**Service Provider**
See Telco.

**SGML**

**SGSN**
Serving GPRS Support Node

**SLC**
Service Logic Controller (formerly UAS).

**SLEE**
Service Logic Execution Environment

**SMS**
Depending on context, can be:
- Short Message Service
- Service Management System platform
- NCC Service Management System application

**SN**
Service Number

**SS7**
A Common Channel Signalling system used in many modern telecoms networks that provides a suite of protocols which enables circuit and non circuit related information to be routed about and between networks. The main protocols include MTP, SCCP and ISUP.

**SSN**
Subsystem Number. An integer identifying applications on the SCCP layer. For values, refer to 3GPP TS 23.003.

**SSP**
Service Switching Point

**Switching Point**
Anything that can send and receive C7 messages.

**System Administrator**
The person(s) responsible for the overall set-up and maintenance of the IN.

**TCAP**
Transaction Capabilities Application Part – layer in protocol stack, message protocol.

**Telco**
Telecommunications Provider. This is the company that provides the telephone service to customers.

**Telecommunications Provider**
See Telco.

**VLR**
Visitor Location Register - contains all subscriber data required for call handling and mobility management for mobile subscribers currently located in the area controlled by the VLR.

**VPN**
The Virtual Private Network product is an enhanced services capability enabling private network facilities across a public telephony network.

**VWS**
Oracle Voucher and Wallet Server (formerly UBE).
XMS

Three letter code used to designate some components and path locations used by the Oracle Communications Network Charging and Control *Messaging Manager* (on page 38) service and the Short Message Service. The published code is *MM* (on page 38) (formerly MMX).